Digital Dictionary by Using Binary Search Algorithm

Project Report

Submitted by:

Ezatullah Amin (2K19/IT/047)

Loem Socheanet (2K19/IT/073)

Mrs. Swati Sharda

DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi- 110042

Acknowledgement

We have taken efforts in this projects. However, it would not have been possible without the kind support and help of many individual. We would like to extend our sincere thanks to all of them.

We are highly indebted to ‘Mrs. Swati Sharda’ for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

We would like to express our gratitude towards our parents and members of DTU for their co-operation and encouragement which help us in the completion of this project.

We thank and appreciations also go to my team member Ezatullah Amin and Loem Socheanet in developing the project and people who have willingly helped us out with their abilities.

Abstract

Digital Dictionary of Computer and Network Engineering can be used in the learning process of computer and network engineering students. Dictionaries are generally book-shaped is hard to carry because of their thick and heavy, now can be accessed anywhere by development of web technology. Digital dictionaries can be accessed through computers, laptops, and cellphones. This is make student easily in learning process. The digital dictionary can not only be used by students of computer and network engineering, but can also by general public, as long as you have the words existing in the dictionaries. The main process in this digital dictionary is the search process. Binary search algorithm used in the search process of binary search algorithm. Binary search algorithm search is applied to word search in this digital dictionary, because this algorithm is intended for sequential data. Program language this is used to build this digital dictionary is c++.

TABLE OF CONTENTS

1. Introduction

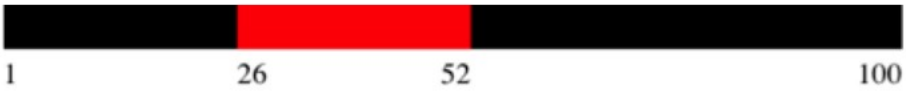
Search is a fundamental process in data processing. The search process is to find a certain value (data) in a set of data of the same type (either base type or form type). The search process is an initial activity related to data processing. In the search process the methods used vary, among others linear or sequential search method, binary search method, direct search method, interpolation search method and hash search method. Of the five available methods, the linear search method (linear or sequential search) is the most commonly used method. The problem that is often encountered in using linear search methods (linear or sequential search) is that if the data sought does not exist, then the search will continue until the last data. This cause linear search to be ineffective applied to large amounts of data. Therefore, the Binary Search method is used to solve that problem. The advantage of the binary search method (Binary Search) rather than the linear search method is the division of the number of elements. In the linear search method (linear or sequential search), the search is carried out directly within the element, while in the binary search method (Binary Search) the search is done by dividing the element into two parts so that the search can be more effective by dividing the search in two directions, so this method is quite efficient. So that the binary search method is very effective applied to large amounts of data.

On the other hand, searching means to find out an item from a list of sorted data. There are a lot of technique for searching. The traditional linear search is the simplest searching technique that has a time complexity of O(n) requiring n comparisons to find out an item in the worst case. On the other hand, one of the efficient search technique is the binary search possessing a time complexity of $O(\log_2 n)$. It will be better if the time complexity of the traditional binary search can be further reduced. From this thinking we have used the concept of indexing to reduce the searching domain and thus to reduce the time complexity of the binary search which is the main target of this project.

Let's look at how to describe binary search carefully. The main idea of binary search is to keep track of the current range of reasonable guesses. Let's say that I'm thinking of a number between 1 and 100, just like the guessing game. If you've already guessed 25 and I told you my number was higher, and you've already guessed 81 and I told you my number was lower, then the numbers in the range from 26 to 80 are the only reasonable guesses. Here, the red section of the number

line contains the reasonable guesses, and the black section shows the guesses that we've ruled out.

In each turn, you choose a guess that divides the set of reasonable guesses into two ranges of roughly the same size. If your guess is not correct, then I tell you whether it's too high or too low, and you can eliminate about half of the reasonable guesses. For example, if the current range of reasonable guesses is 26 to 80, you would guess the halfway point, (26 + 80) / 2(26+80)/2left parenthesis, 26, plus, 80, right parenthesis, slash, 2, or 53. If I then tell you that 53 is too high, you can eliminate all numbers from 53 to 80, leaving 26 to 52 as the new range of reasonable guesses, halving the size of the range.



We could make that description even more precise by clearly describing the inputs and the outputs for the algorithm and by clarifying what we mean by instructions like “guess a number” or “search a word” and "stop."

The algorithm will be effective if run by a processor (processor). The processor can be human, computer, robot, machine etc. The processor reads every instruction in the algorithm and then does it. According to Les Goldschlager [GOL88], a processor must: Understand each step in the algorithm. Work on operations that correspond to these steps. Search is a fundamental process in data processing. The search process is to find a certain value (data) in a set of data of the same type (either base type or form type). For example, to change (update) certain data, the first step that must be done is to find the existence of data in the collection. If the data sought is found, the data can be changed in value with new data. The same initial activity is also carried out in the process of adding new data. The process of adding data begins by finding out whether the data to be added is already in the collection. If it already exists and considers there is no duplication of data, then the data does not need to be added, but if it doesn't already exist, then add it [9] 3. Result and Analysis 3.1. Implementation System After the system is analyzed and designed in detail, then go to the implementation stage. Implementation is the stage

of placing the system so that it is ready to operate. The implementation aims to configure the design into the system.

2. Methodology

There are 4083 words that we can search for in this dictionary and the Binary Search Algorithm successfully finds every word that is searched if the word is available in the dictionary database. The search process on the “Digital Dictionary using Binary Search Algorithm” works well. This Digital Dictionary can be easily used by users who want to find computer related vocabulary. We can also access the history by giving some input to the program. If we have previously searched for words it will appears there. And give us information related to the words that have been sought out before. If the user wants to close the dictionary there are two ways to perform this action either by windows closing buttons or by inserting a value to the program. The principle of searching by dividing data on two parts inspires binary search methods. Data stored in arrays must be sorted. To ease discussion, then the array elements are sorted down (descending). In the search process, two array indices are needed, that are smallest index and the largest index. The smallest index is assumed as the index of the left end of the array and the largest index is the index of the right end of the array. The terms "left" and "right" are expressed by imagining array elements stretched horizontally. for example, the left index is i and the right index is initially initializing i with i and j with n.

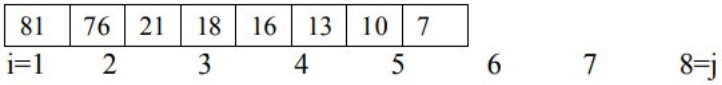
Step 1: Divide into two array elements in the middle element. The middle element is an element with an index k = (i + j) div 2. (middle element, L[k], divide the array into two parts, that are the left side L[i..j] and right side L [k + 1 ..j])

Step 2: Check if L[k] = x. If L[k] = x, search is complete, because x has been found. But, if L[k] ≠ x, must be determined whether the search will be carried out on the left or right. If L[k]x, the search is carried out again in the right-hand array.

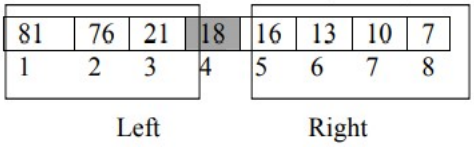
Step 3: Repeat to step 1 until x found or i > j (that is, the array size is zero). Example search element with binary search method:

2.1 Illustrate binary search:

For example, given an L array with eight elements that have been sorted down as below:



a. For example the element sought is x = 18. Step 1: Middle element index k = (1 + 8) div 2 = 4 (shaded element).



Step 2:

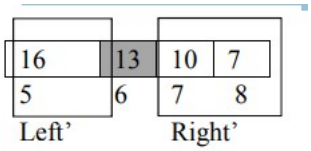
Compare it, if L [4] = 18? Ya! (x found, the search process is complete)

b. For example, the element sought is x = 16.

Step 1:

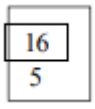
i = 1 and j = 8

Middle element index k = (1 + 8) div 2 = 4 (shaded element)



Compare it: If L [6] = 16? No! Must be determined whether the search will be carried out on the left or right side by checking as follow:

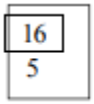
Compare: If L[6] > 16 ? No! Do search on the left side array with i = 5 (fixed) and j = k – 1 =5.



Step 1 :

i = 5 and j = 5

Middle element index k = (5 + 5) div 2 = 5 (shaded element)



Step 2 :

L[5] = 16 ? Ya! (x found, search proses complete)

2.2 The processing in this project:

When you run this program first of all it will show a cover of our project then it will show names of our team member. After that, another window will show with some options for user to choose. There are:

1. Search a word
2. See the history
3. Exit

2.2.1 Search a word

We use binary search here for searching a word in this dictionary. First of all, whenever you input a word, it will check the middle word in this dictionary and then if your word is larger than the middle once it will compare with the lager haft. But if your word is lower than the middle once it will compare with the lower haft. And it will repeat these steps again and again till your word is found.

For example: You choose option “Search” then you need to input a word. Assume that you input “cathode”.

There are 4083 words in our dictionary so in the binary search algorithm

- Step1: It finds the middle index of our array first. By taking first index of our array that we have declare as “low” and the highest index which is declare as “high” plus with each other than divide by 2.
  mid=(low+high)/2;
  mid=(0+4082)/2;
  mid=2041;

  word1[mid]= “impromptu”

- Step 2: After we found the middle index of this array, we will compare the word “cathode” and “impromptu”. Now we see that our word is less than “impromptu” so it will set the ‘high’ variable = mid – 1 (high = 2041 – 1=2040); then we will find the middle value again by (mid= (low+high)/2).

So now we get our mid = (0+2040)/2
mid=1020;
word1[mid]= “creed”;
so “cathode” < “creed”

- Step 3: It will repeats step 2 again.
  so we get:
  high = mid -1;
  high = 1019;
  mid=(low+high)/2;
  mid=(0+1019)/2;
  mid=560;
  word1[mid]= “breach” is < “cathode”
- Step 4 :
  low=(mid+1);
  mid=(low+high)/2;
  mid=(561+1019)/2;
  mid=790;
  word1[mid]= “competitor”> “cathode”
- Step 5 repeat step 3 agin:
  high=mid-1;
  high = 789;
  mid = (low + high)/2 ;
  mid = (561+789) /2 ;
  mid= 675;
  word1[mid] = “censer” > “cathode”
- Step 6: it will repeat step 5;
  high = mid -1 ;
  high = 674;
  mid = (low + high) /2;
  mid = (561+674)/2;
  mid = 618;
  word1[mid]= “calculable” < “cathode”;
- Step 7 : it will repeat step 4;
  low = mid +1 ;
  low = 619;
  mid = (low + high)/2;
  mid = (619+674)/2;

mid = 647 ;
word1[mid] = “captivate” < “cathode”

- Step 8 will repeat step 7 again;
  low=mid + 1;
  low = 648;
  mid = (low + high)/2;
  mid = (648 + 674) / 2;
  mid = 661;
  word1[mid] = “casual” < “cathode”
- Step 9 will repeat step 8 again;
  low = mid + 1;
  low = 662;
  mid = (low + high) /2;
  mid = (662+674)/2;
  mid = 668;
  word1[mid] = “catholicity” > “cathode”
- Step 10 will repeat step 6;

  high = mid -1;

  high = 667;

  mid = (low + high) /2 ;

  mid = (662 + 667)/2;

  mid = 665;

  word1[mid]= “catastrophe” < “cathode”

- Step 11 :
  low = mid + 1;
  low = 666;
  mid = (low + high)/2;
  mid = (666+ 667)/2;
  mid = 667;
  word1[mid] = “catholicism” > “cathode”
- Step 12:
  high = mid -1 ;
  high = 666;
  mid = (low + high) /2;

mid = (666+666)/2;
mid = 666;
word1[666]= “cathode”== “cathode”

Finally our word “cathode” is found in index 666.

2.2.2 To see the history

When you choose option “History”. First of all I have created a file in this program to store all the words that we have search in this dictionary by using a function “ifstream file("history.txt",ios::out)”.

Then I put a while loop for adding all the words that we have searched. After that all the word that we have search will automatically save into our file. So whenever we want to see our history just print the word in our file out and that’s it for history part.

For example:

When we searched a word “cathode” like above then it will be store the word “cathode” in the txt file. Then whenever we want to see the history of the word that we have searched then it will appear on the screen.

2.2.3 Exit

Option “Exit” it is the third option in our program. It works for exiting the program when you want to exit this digital dictionary. It is really easy for the user to exit the program by just pressing number ‘0’ or any other number out of the options that have shown on the screen.

3. Result and Analysis

3.1 Implementation system

After the system is analyzed and designed in detail, then go to the implementation stage.

Implementation is the stage of placing the system so that it is ready to operate. The implementation aims to configure the design into the system. This is so that users can provide input to system developers.

3.2 Digital Dictionary Display

Cover page

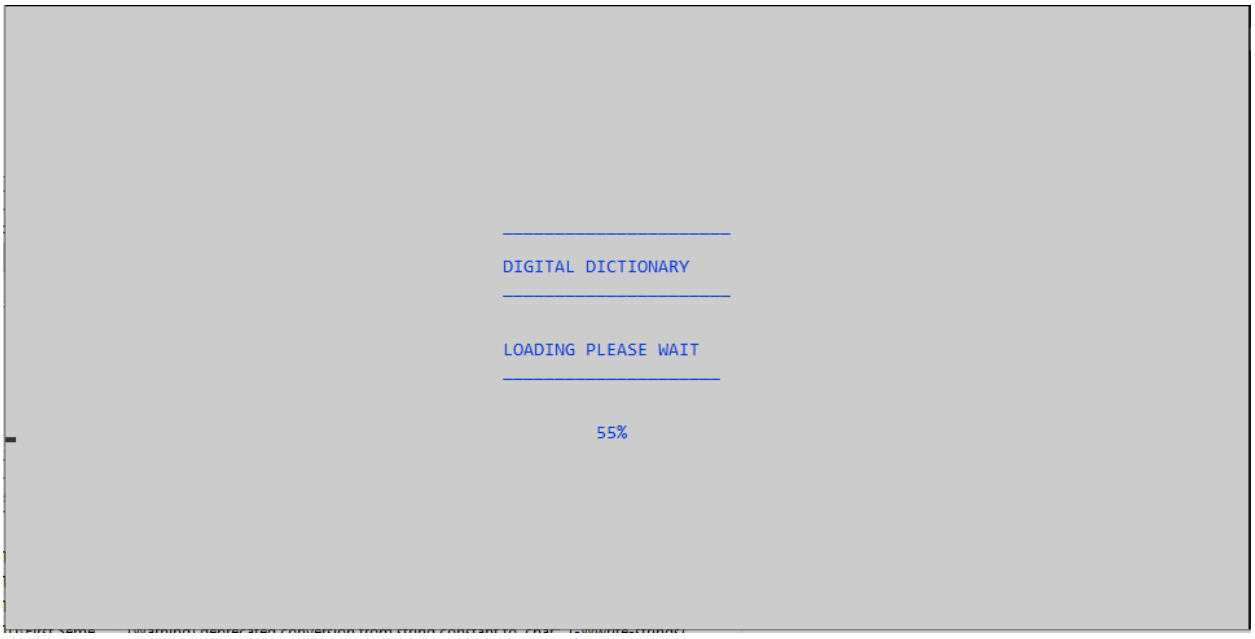


Figure 1. Cover Page (1)

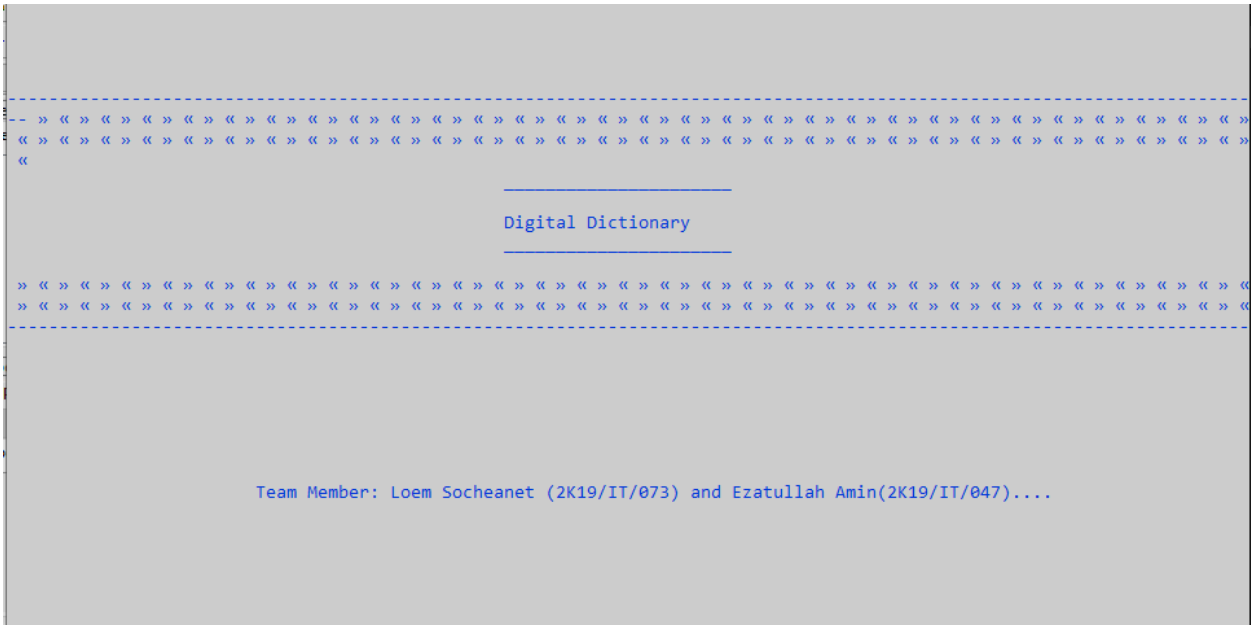


Figure 2. Cover Page (2)

Main Page

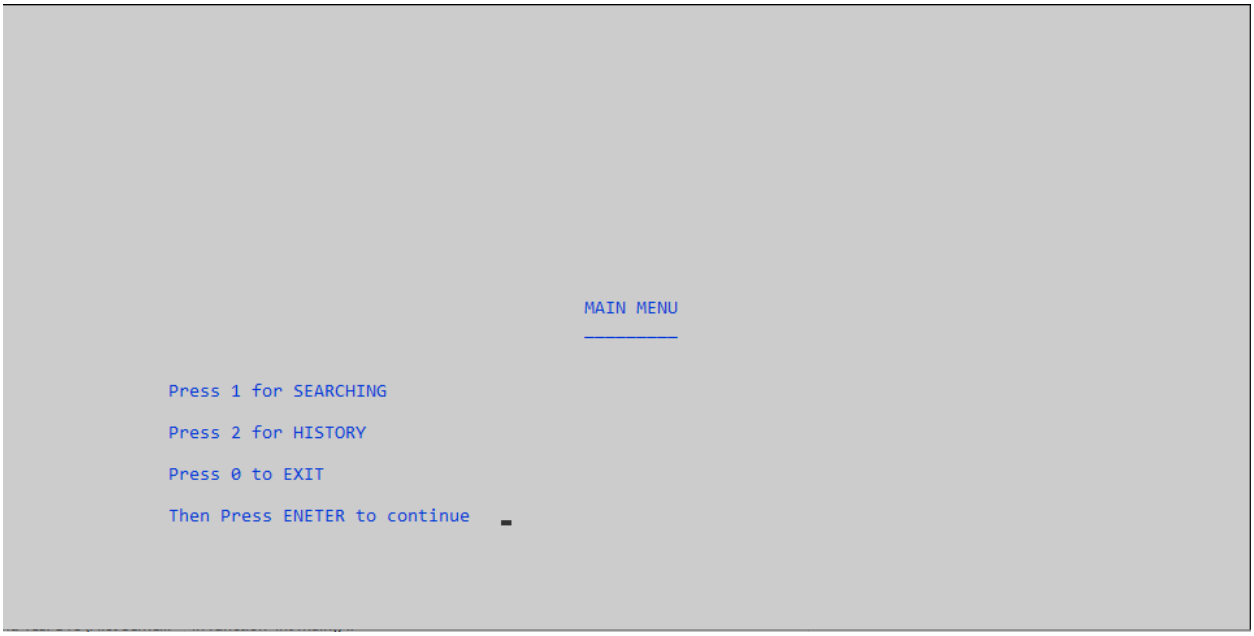


Figure 3. Main Page

Searching Page

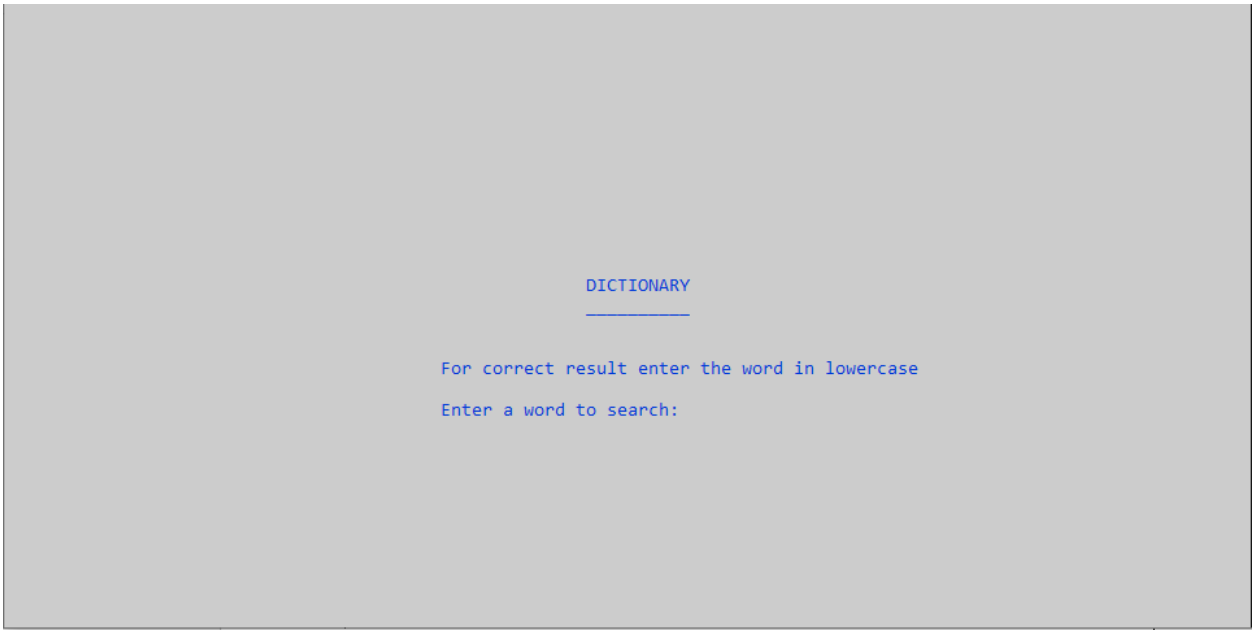


Figure 4. Searching Page

Result Page

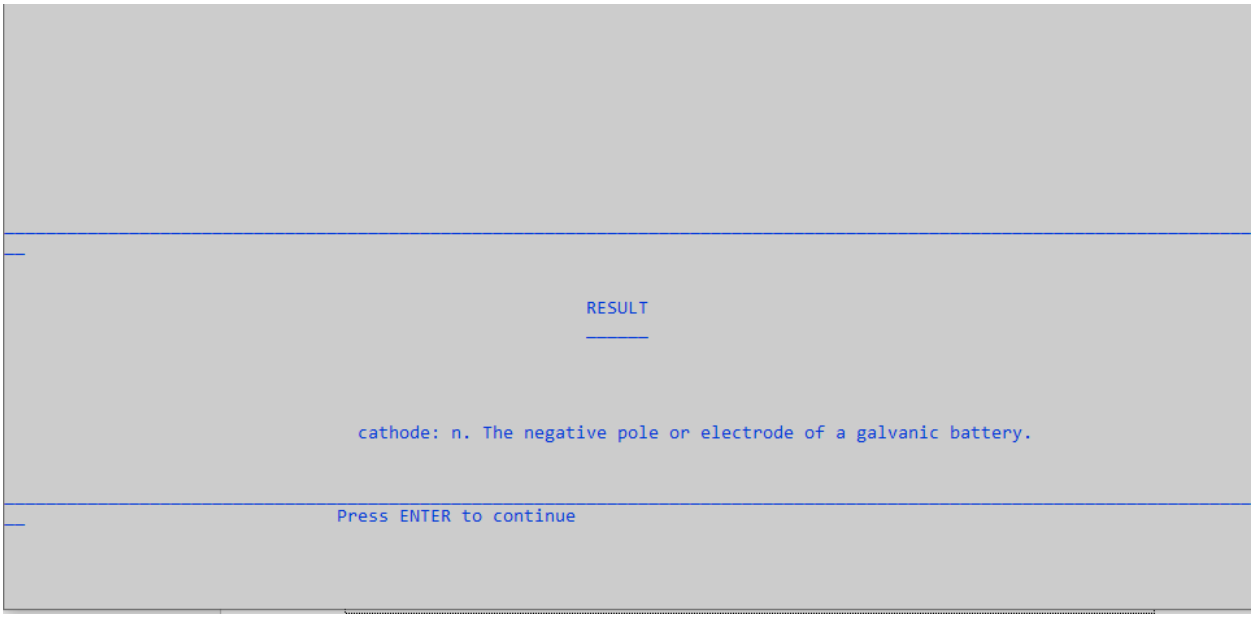


Figure 5. Result Page

History Page

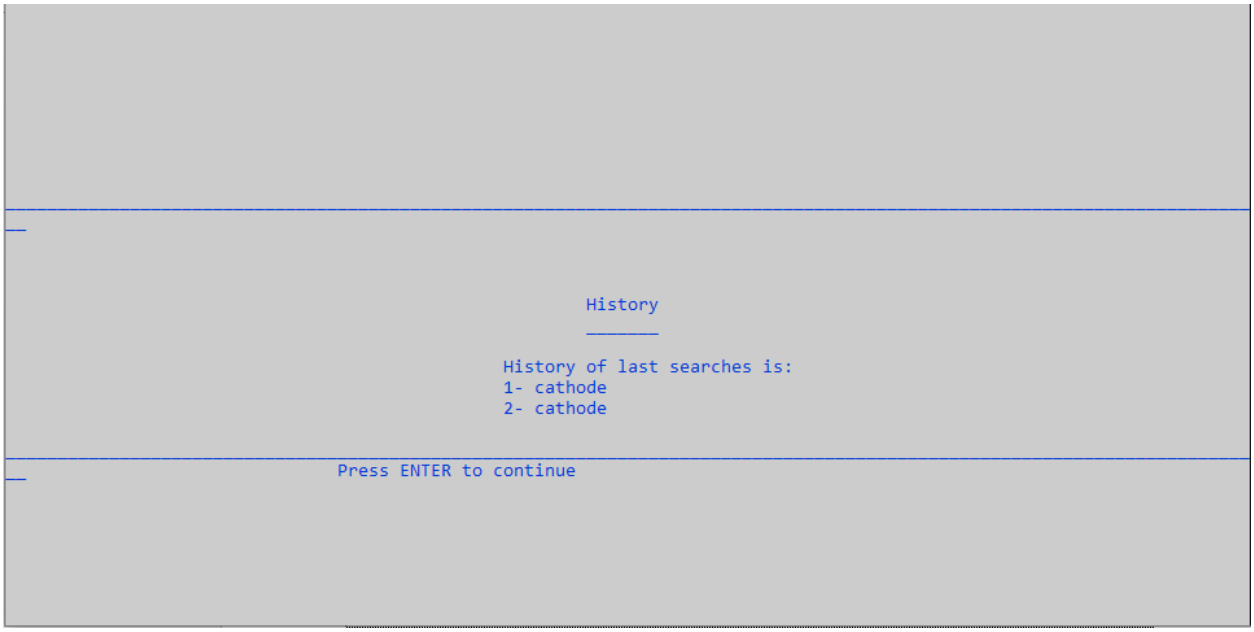


Figure 6. History Page

4. Conclusion

From the results of the discussions that have been carried out, it can be concluded as follows:

a. The search and history process on the “Digital Dictionary by Using Binary Search Algorithm” C++ built using binary search algorithms works well.

b. The Binary Search Algorithm successfully finds every word that is searched if the word is available in the dictionary.

c. The basic principle of the binary search algorithm is to repeat the search space repeatedly until data is found or until the search space cannot be shared (data may not exist).

d. The purpose of searching using a binary search algorithm is to reduce the number of operations that must be compared between the data sought and data in the database, especially for large amounts of data.

e. This digital dictionary is built to facilitate students, especially computer and network engineering students in finding foreign computer vocabulary without having to carry a dictionary in book form everywhere.

f. This digital dictionary is also made to make it easier for people to find computer vocabulary that they don't understand.

g. This Digital Dictionary can be easily used by users who want to find computer vocabulary.

5. References

[1] Digital Dictionary Using Binary Search Algorithm (Web built)

Riski Muhamad Fitrian, Insan Taufik, Muhammad Sabir Ramadhan, Neni Mulyani, Jeperson Hutahaean, Arjon Samuel Sitio and Hengki Tamando Sihotang. (https://iopscience.iop.org/article/10.1088/1742-6596/1255/1/012058/meta) Form Google Scholar.

[2] How to make Dictionary in C++. From Organic Tech.

(https://drive.google.com/file/d/1h8QkUVyC0PEjWgv776WeGbETNqPAuizu/view)

[3] GeekforGeeks (https://www.geeksforgeeks.org/)

[4] Wikipedia (https://www.wikipedia.org/)